A Location Service Mechanism for Position-Based Multicasting in Wireless Mobile Ad hoc Networks*

Yoav Sasson David Cavin André Schiper
{yoav.sasson,david.cavin,andre.schiper}@epfl.ch

École Polytechnique Fédérale de Lausanne (EPFL)
1015 Lausanne, Switzerland

Technical Report IC/2003/29

Abstract

In this paper we propose a novel location management scheme tailored for multicasting in Mobile Ad-hoc Networks (MANETs). We furthermore propose AMDLM, a location-based multicast algorithm relying on the location management service. Such an approach avoids fragile data structures such as trees or DAGs to manage multicast groups, without reverting to more reliable, yet overhead-prone mesh-based algorithms. AMDLM additionally enables us to derive analytical bounds due to its location-based nature.

1 Introduction

Mobile ad hoc networks (MANETs) are self-organizing mobile wireless networks that do not rely on a preexisting infrastructure to communicate. Nodes of such networks have limited transmission range, and packets may need to traverse multiple other nodes before reaching their destination. Research in MANETs was initiated 30 years ago by DARPA for packet radio projects [JT87], but has regained popularity nowadays due to the widespread availability of portable wireless devices such as cell phones, PDAs and WiFi / Bluetooth enabled laptops.

Multicasting provides a means for multipoint communication by enabling applications to seemingly communicate with groups of nodes. Traditionally a well suited tool for collaborative applications, multicasting is especially useful in ad hoc networks where tasks may be carried out by groups of nodes. Due to scarce bandwidth, varying network connectivity and frequent topology changes caused by node mobility and transient availability, routing algorithms tailored for wired networks will not operate well if directly transposed to MANETs. All the more so with multicasting, which adds to the difficulties of unicast routing the complexity of maintaining and handling dynamic multicast group membership changes.

Since no fixed infrastructure of servers is assumed in MANETs, it is useful to devise a scheme through which various services offered within the network may be located. With the availability

*The work presented in this paper was supported by the National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), a center supported by the Swiss National Science Foundation under grant 5005-67322.

of such location management services, it is tempting to adapt and exploit them for storing routing information. By storing the geographic location of nodes in designated location servers in the network, it is possible to introduce a new family of routing algorithms that may potentially perform better [CBW+02] than the traditional approach of discovering and maintaining end-to-end routes.

In this paper we present a novel MANET location service for multicasting based on an extension of the DLM scheme [XLN01]. Dedicated servers are distributed throughout the network in order to store the geographic location of multicast group members. Coupled with an underlying geographic forwarding layer (e.g. [KK00, Sto02]), the solution offers an alternative approach for multicast routing. Among the benefits of location-based multicasting are: reduced overhead, increased robustness to mobility, fault-tolerance and the ability to derive analytical results. There exists several location-based algorithms for *unicast* routing ([BCSW98, LJC+00, XLN01, WS01, CLP+02, HB03]). [MFWL03] extend unicast position-based routing for multicasting. The authors offer a generalization for the routing aspect and assume that the position of the destination(s) is known in advance through a location service. Due to node mobility and dispersion of multicast node members, we claim that location services designed for unicast routing are not exploitable as such for multicasting. The contribution of this paper is to devise a novel location management scheme adapted for multicasting in MANETs. We furthermore present *AMDLM* (*Adaptive Multicast Distributed Location Management*), a location-based *multicast* algorithm built on top of the location service[1].

The remainder of the paper is organized as follows. The next section provides an overview of other works that address multicasting in MANETs. In Section 3 we present the DLM [XLN01] location management scheme, which serves as a basis for our multicast algorithm. Section 4 describes the necessary modifications in order to extend DLM for multicasting, followed by a discussion about the limitations of mere straightforward extensions. Section 5 presents *AMDLM*, a novel location based multicast algorithm. In Section 6 we undertake an analytical study of the algorithm, followed in Section 7 by a qualitative comparison between *AMDLM* and the other popular approaches for MANET multicasting. We finally conclude and describe future work in Section 8.

2 Related Work

Over the past years there have been numerous multicast algorithm proposals for MANETs. In this section we present the most representative for each approach, classified in Table 1.

As with unicast routing [HXG02], multicast routing comes in *proactive*, *reactive*, or a combination of the two flavors (*hybrid*). Reactive algorithms present reduced maintenance overhead by maintaining state information only when a multicast session is active. The drawback is decreased responsiveness. Proactive algorithms react faster since multicast routing information is readily available, but at the price of introducing high overhead for maintaining multicast group structure even when no multicast session is active. The hybrid approach aims at obtaining a satisfactory balance among the characteristics of both methods by limiting the scope of the proactive procedures to the local neighborhood of nodes and implementing reactive procedures for longer distances.

Various algorithms rely on different data structures to manage multicast group membership. Due to the highly dynamic and everchanging topology of MANETs, solutions that offer multiple routes through more robust data structures perform better. Therefore, mesh-based solutions generally outperform tree-based solutions due to the availability of alternative paths, which in turn tend

[1]Location-based multicasting may be confused in the MANET community with *geocasting* [YKV99]. Whereas for geocasting nodes join and leave groups by entering and leaving geographic regions, multicasting enables nodes to join and leave groups at any time, regardless of their location. *AMDLM* provides the latter service.

to perform better than directed acyclic graph (DAG) solutions. In extreme cases of high mobility and frequent multicast group membership changes, basic flooding still remains the best performing multicast algorithm [HOTV99]. Performance comparison studies of MANET multicast protocols may be found in [LSH+00, OTV01, KC02].

Our location-based approach for multicasting, *AMDLM*, differs from previous MANET multicast algorithms by relying on a location service composed of dedicated nodes distributed across the network[2]. Communication between these nodes is provided by an underlying geographic forwarding routing mechanism. Therefore, to the contrary of the tree, DAG or mesh approach, no multihop data-structure needs to be maintained, hence the distinction between the *physically connected* and *logically connected* data-structures in Table 1. Physically-connected data-structures are generally more vulnerable to frequent link-breakages that occur in MANETs.

	Physically connected				*Logically connected*
	Tree	*Mesh*	*Hybrid Tree-Mesh*	*DAG*	*Location Based*
Proactive		CAMP [MGLA01] FGMP [CGZ98]	AMRoute [LTM99]	AMRIS [WT99]	
Reactive	MAODV [RP99] ADMR [JJ01] ABAM [TGB00] LAM [JC98] OLAM [BCST00]	Flooding ODMRP [LCG99]	MCEDAR [SSB99]		AMDLM
Hybrid	MZR [DS01]				

Table 1: MANET Multicast Algorithms

3 The Distributed Location Management Scheme (DLM)

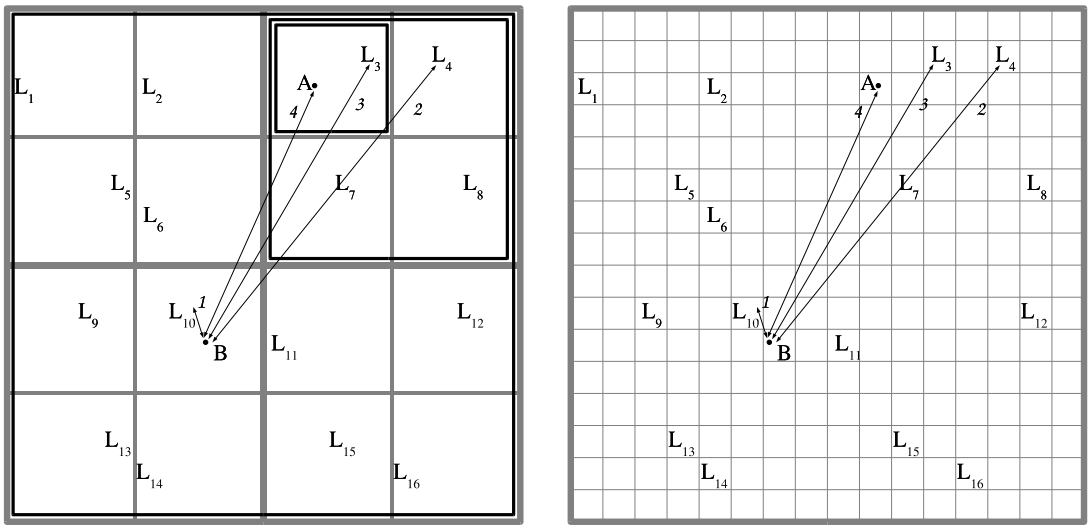
DLM [XLN01] is a location management service for MANETs tailored for *unicast* routing, which addresses the shortcomings of GRID [LJC+00]. As with GRID, DLM partitions the mobile node deployment region into a hierarchical grid with squares of increasing size, as shown in Figure 1(a). The location service is offered by location servers assigned across the grid, storing node location information. DLM assumes a *uniform* distribution of the location servers. The server density is a parameter that may be adapted to better suit the characteristics of the network. *To the contrary of GRID, location servers in DLM are not directly nodes, but regions in the grid.* Nodes that happen to be located in these region offer the location service. This solution increases DLM's robustness to mobility. The selection mechanism for the predetermined regions is carried out through a *hash function*, which maps node identifiers to region addresses.

DLM distinguishes between a *full length* address policy and a *partial length* address policy. Under the full length address policy, location servers store the precise position of nodes. When nodes change regions due to mobility, it is necessary to update all location servers. Under the partial length address policy, the accuracy of node location information stored at the location servers increases along with the proximity of the location servers to the nodes. To the contrary of the full length address policy, several queries are necessary to locate a node. Nevertheless,

[2] [MFWL03] offers a multicast generalization of position-based routing. The paper however assumes a functioning location service from which the position of the destination(s) may be retrieved. The main contribution of our paper is specifically a solution for such a location service.

the partial addressing scheme offers overall increased performance, since it reduces the scope and frequency of location server updates due to node mobility. Indeed, only the location servers affected by the distance travelled by the nodes need to be updated. We therefore consider the partial length address policy whenever we refer to DLM in this paper.

Figure 1 illustrates an example of how a node location query is carried out in DLM. Figure 1(a) depicts the location server hierarchical partitioning, which is an abstract overlay above the full grid in which the nodes evolve (Figure 1(b)). A location server is responsible for its entire region, which may not be within single-hop communication reach. Node B desires to find the location of node A. B will first query location server $L10$, which is A's location server in the same region as B. $L10$ will reply to B with information about the quadrant A belongs to. B will now contact a location server for A in that quadrant, e.g. $L4$. $L4$ will similarly reply to B with the smaller subregion containing A. The process continues until B eventually contacts a location server that knows A's exact position, $L3$ in our example.



(a) Location Server Partitioning Representation

(b) Full Grid Representation

Figure 1: Node Location Query under DLM's Partial Address Policy

4 DLM and Multicast

In this section we examine the straightforward modifications required for DLM to offer a multicast service.

4.1 MDLM: Extending DLM for Multicast

DLM scheme relies on a *hash function* for assigning and locating servers responsible for storing node location information. The assignment is essentially based on the identifier of the node we wish to locate. DLM may be extended to offer a multicast service by replacing the node ID parameter by a multicast group ID. Furthermore, location servers will now store a *set* of links to regions that contain multicast group members. The rest of the algorithm remains similar. In the remainder of the paper, we denote by *MDLM* the straightforward extention of DLM for the multicast operation.

Figure 2 illustrates an example of a node B multicasting to a group G composed of members $G = \{A, A', A''\}$, with the multicast extensions brought to DLM. B must first obtain the location of the group members. B will first query location server $L10$, which is group G's location server in the same region as B. $L10$ will reply to B with a set of quadrants that contain group members. B will then contact a location server for G in each one of the quadrants returned, i.e. $L4$, $L12$ and $L13$. These location servers will similarly reply to B with the smaller subregions containing members of G. The process continues until B eventually contacts the location servers that know the exact position of each group member, $L3$, $L13$ and $L16$ in our example. B may now send a message to A, A' and A''.

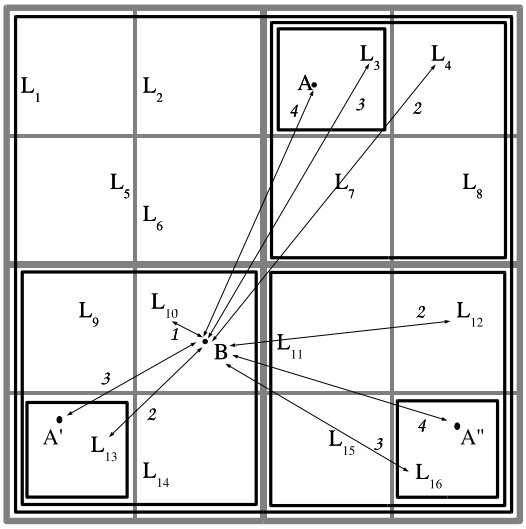


Figure 2: DLM Multicast Extension

There are also trivial modifications that may be brought to DLM's approach in order to greatly enhance performance. In particular, instead of location queries being sent back and forth between a requesting node and the location servers until location information is gathered, a multicast may be sent and routed through the location servers themselves. Multicast messages will be forwarded along the location servers until they reach their destination. Among the benefits are reduced latency, increased reliability and robustness to mobility, since messages may still reach a moving target.

4.2 Discussion

The solution presented in Section 4.1 to transform DLM into a multicast service (MDLM) is rather straightforward but has nevertheless drawbacks in terms of performance, overhead and scalability.

In the case of a uniform distribution of *multicast group members*, MDLM may be satisfactory. This is however not the case with a non-uniform geographic distribution of the group members, a situation that may be very frequent. If group members are not uniformly distributed, having a uniformly distributed set of location servers is not optimal in terms of cost to maintain the location information (in regions void of group members) when nodes move, join, or leave the group.

Ideally, the presence or absence of location servers in a region should dynamically adapt to the presence or absence of group members in that region. In the rest of the paper we present a solution that has this property. We also discuss how nodes *join* and *leave* a group and the handling of node

mobility.

5 The Adaptive Multicast Distributed Location Management Algorithm (AMDLM)

5.1 Design Choices for Efficient Location Management Multicasting

The property we most desire for a location-based MANET multicast algorithm is to minimize the number of required location servers without harming overall performance. This may be achieved through maintaining a higher concentration of location servers around multicast group members, requiring a *dynamic* assignment and adaptation of the location servers. Although multicast group members and nodes in their proximity will have privileged access to multicast group membership information, nodes far from group members can still multicast, but with a higher average cost.

An additional desirable property is the independence of the core multicast algorithm from a particular location server placement scheme. This is achievable by isolating key hash function properties from a specific implementation. The benefit will be a more flexible location-based multicast algorithm, since the hash function responsible for placing the location servers may be chosen to better suit a particular MANET topology.

5.2 Model

The model for *AMDLM* is similar to GRID [LJC+00] and DLM [XLN01]. Wireless nodes evolve in a geographic area partitioned into a hierarchical grid with squares of increasing size. The smallest region contains one square and is referred to as the $level_0$ region. Four $level_0$ regions form a $level_1$ region and so on, as shown in Figure 3. Regions do not overlap, so a $level_k$ region belongs to exactly one $level_{k+1}$ region (and thus nodes belong to exactly one region of each size). We further assume as with position-based routing algorithms that nodes are aware of their location through a positioning system such as GPS. Since the grid is static and predetermined, nodes know of their current region within the grid, as well as its boundaries. Nodes sharing the same $level_0$ region are called *neighbors*. All neighbors have knowledge of each other, even though they may not be within a 1-hop communication range (i.e. through flooding or token passing)[3]. Finally, similarly to DLM, we assume node density and distribution such that statistically over time, at least one node will be present per $level_0$ region (the grid dimensions may be chosen accordingly).[4]

Given two regions reg_1 and reg_2, we denote by $reg_1 \cup reg_2$ the geographic region that corresponds to the union of reg_1 and reg_2, by $reg_1 - reg_2$ the geographic region such that reg_2 is removed from reg_1; moreover $reg_1 \subset reg_2$ is true if reg_1 is included in reg_2.

5.3 Logical Servers and their Placement

Given $k \geq 0$, we define R_k as any $level_k$ region. For a node n_i, $R_k^{n_i}$ is the region R_k such that $n_i \in R_k$. *Logical servers correspond to $level_0$ regions*: nodes happening to be in a given $level_0$ region that correspond to a logical server, participate in the service. Note that, as already explained, not every $level_0$ region is necessarily a logical server.

[3]The size of $level_0$ regions may also be chosen as to be fully covered by the transmission range of the nodes. In this case, the term *neighbors* still refers to two nodes sharing the same $level_0$ region and **not** to any two nodes within communication range.

[4]This assumption does not mean uniform distribution of group members!

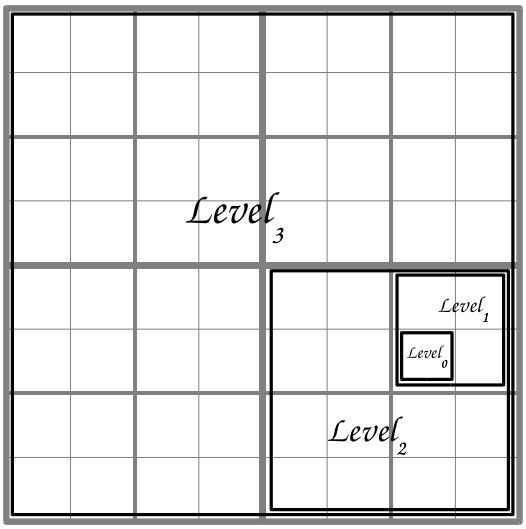


Figure 3: Grid Partitioning Scheme

The logical servers are denoted by S. More precisely, we denote by $S_k^{n_i}(G)$ the logical server(s) in a region R_k for node n_i and group G. $S_k^{n_i}(G)$ will simply be denoted $S_k^{n_i}$ in the rest of the presentation. We now present the placement rules of our logical servers for a given group G.

Property 1 (Partitioning Rule)
$k = 0$: For each node n_i member of G, there exists exactly one logical server of level 0 (denoted by $S_0^{n_i}$), which is region $R_0^{n_i}$.
$k > 0$: For each level $k > 0$ and for each node n_i, there exists exactly four logical servers of level k (denoted by $S_k^{n_i}$). Moreover, there is exactly one of these four servers S_k in each of the four sub-regions region $R_{k-1}^{n_i}$ of region $R_k^{n_i}$.

Figure 4(a) illustrates the partitioning rule.

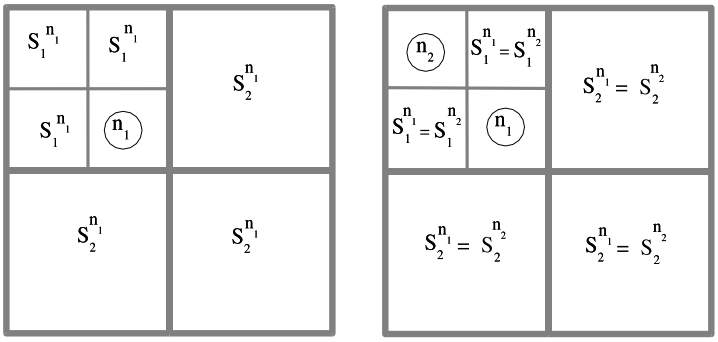
Property 2 (Sharing Rule)
$\forall k > 0$ and for two nodes n_i and n_j member of the same group, if $R_k^{n_i} = R_k^{n_j}$, then n_i and n_j share the same level k server(s), i.e. $S_k^{n_i} = S_k^{n_j}$.

Figure 4(b) illustrates the sharing rule.

5.4 Locating Logical Servers

Nodes determine the geographic position of the *logical servers* by means of a *hash function*. For a given multicast group G and level $k > 0$ (i.e., for group G and region R_k), the hash function returns the four $level_0$ regions corresponding to the four logical servers in R_k for group G. Note that these $level_0$ regions are not necessarily logical servers, since the presence of logical servers depends on the presence or not of members of G in region R_k. So a node n_i — located in the level 0 region of address a [5] — wanting to multicast a message to group G first needs to find the smallest $k \geq 0$ such that there is a logical server of level k in the R_k region of n_i. This is done using the hash function $\eta(k_{max}, k, G, a)$ (see Appendix B), which returns the potential logical server for G in region R_k of n_i.

[5] Each R_0 region can be uniquely identified with an address (see Appendix B.1).



(a) Property 1: Partitioning Rule

(b) Property 2: Sharing Rule

Figure 4: Hash Function Properties

5.5 Joining and Leaving Groups

We now describe how nodes join and leave a group. The corresponding pseudo-code may be found in Appendix C.2.

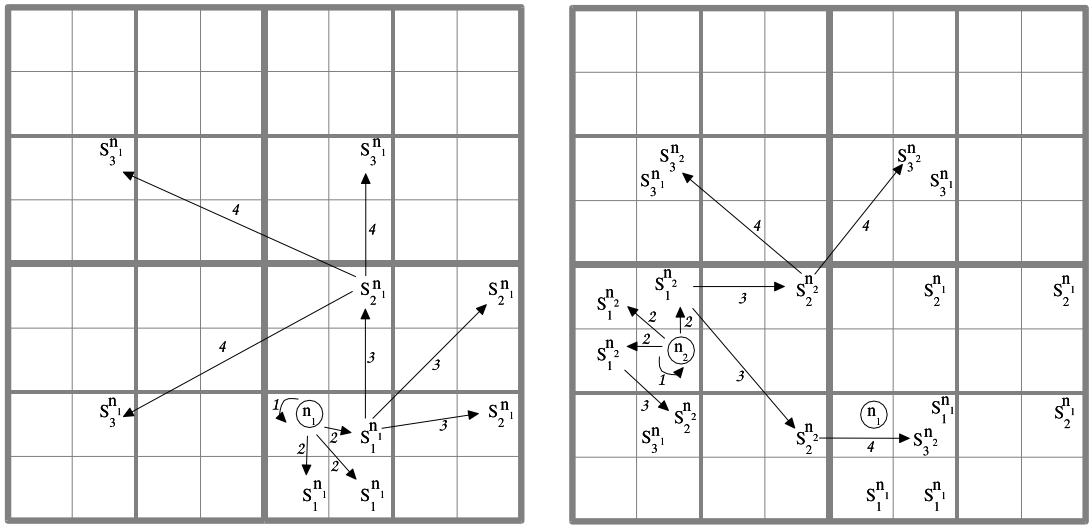
Let node n_1 want to join a new group G (Figure 5(a)). n_1 first issues a `join(G)` request to its $S_0^{n_1}$ (located in the same R_0 region than n_1), which will add an entry for n_1 as belonging to G. Then $S_0^{n_1}$ will in turn forward the *join* to the three neighboring $S_1^{n_1}$ servers, activating them $level_1$ *logical servers* for group G. Activating means that each $S_1^{n_1}$ stores a reference to n_1's $level_0$ region. Through the hash function η, the $S_1^{n_1}$ servers in turn forward the `join(G)` request to three $S_2^{n_1}$ servers. As formally described in Algorithm 5 of Appendix C.2, we may note that requests to $S_{k+1}^{n_1}$ servers are forwarded by the geographically closest $S_k^{n_1}$ servers. As before, each $S_2^{n_1}$ stores a reference to the corresponding $S_1^{n_1}$ from which it received the `join(G)` request. This procedure is repeated on each level until the maximum level is reached. Note that Algorithm 1 (Appendix C.2) specifies that *join* queries are forwarded to upward logical servers only upon receiving a *join* query within the region for the first time, so that only one reference is needed by logical servers S_k although many G group members are in a R_k region.

The leave operation is the reverse of the join operation (see Appendix C.2).

5.6 Multicasting

We illustrate the principle of AMDLM on two examples, see Figures 6(a) and 6(b). The complete pseudo-code of the algorithm may be found in Appendix C.2. In Figure 6(a), the source is member of the group to which it desires to multicast. Figure 6(b) illustrates a case where the source does not belong to the group. We now further define $\overline{S}_k^{n_i}$ as being the $level_k$ location server located in n_i's $level_{k-1}$ region. For sake of clarity, only the logical servers participating in the multicast examples are represented in the figures.

In Figure 6(a), node n_2 desires to multicast a message to group G, containing two members n_1 and n_2. To do so, it first delivers the multicast message to itself (since n_2 is its own $S_0^{n_2}$ server), as well as to $\overline{S}_2^{n_2}$ server. Through Property 2 of Section 5.3, $\overline{S}_2^{n_2}$ knows of any group member within region $R_2^{n_2}$. Since in our case there are no members within $R_2^{n_2}$, the multicast request is forwarded to $\overline{S}_3^{n_2}$. Since $\overline{S}_3^{n_2}$ has an entry for a group member somewhere in a neighboring R_2 region (because $\overline{S}_3^{n_2}$ is also $S_3^{n_1}$), it will forward the request to the S_2 server it references, i.e. $S_2^{n_1}$. Upon receiving the request, $S_2^{n_1}$ will do the same. The procedure is repeated until the multicast message arrives



(a) Node n_1 joining group G

(b) Node n_2 joining group G

Figure 5: Join Operation

at destination (n_1).

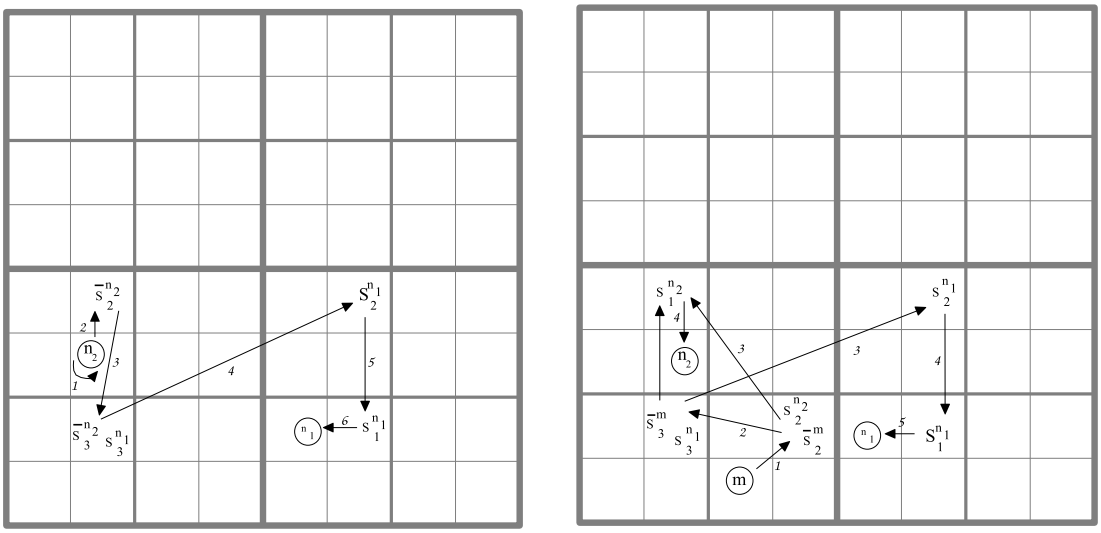
To the contrary of other MANET multicast algorithms mentioned in Section 2, multicasting by nodes not belonging to the intended destination group is not more complex. In Figure 6(b), node m desires to multicast a message to a group G, composed of nodes $\{n_1, n_2\}$. Since it does not belong to the group, m directly forwards the request to $\overline{S}_2^m$, which happens to be one of n_2's S_2 servers. While n_2 will now be forwarded the multicast message, $\overline{S}_2$ has still to forward the request to $\overline{S}_3^m$[6]. The reason being that if there are nodes to be reached in the neighboring R_3 regions, $\overline{S}_3^m$ will surely reference one of their S_2 servers. Indeed, in our case $\overline{S}_3^m$ is also $S_3^{n_1}$. The message will be forwarded to n_1 as described in the previous example.

5.7 Mobility

We consider mobility of multicast group members. Mobility is handled as a *join* operation followed by a *leave* operation, with a *level* parameter (see Appendix C.2). The *level* represents the highest region level the node has crossed during its journey. Straightforward for the node to compute, it allows to bound the propagation of the *leave* request. Consider a node n_i changing $level_2$ regions. No action is taken when a n_i starts moving [7]. Upon reaching its destination (i.e., node speed passes back below a given threshold), n_i rejoins G through a `join` request, with $level = 2$ passed as a parameter. In a second stage, n_i must notify its original $level_0$ region that it has left by sending to it a bounded *leave* request. Finally, the original $level_0$ region propagates the *leave* message to remove n_i's stale entry from the appropriate logical servers. Bounded `join` and `leave` are required since no assumption may be made on their order of reception at the logical servers. Indeed, if the `leave` is received before the `join`, logical servers will confuse mobility with the normal case of a node desiring to leave a group. The *level* parameter solves this problem from preventing the `leave` (and afterward the `join`) from being wrongly propagated higher up in the logical server hierarchy.

[6] since $k < k_{max}$ in Algorithm 3.

[7] While n_i is moving and before any logical server has been updated, messages may be forwarded to it by following its *trail* — nodes n_i encounters during its move store a forwarding pointer to it, à la [LJC+00].



(a) Node n_2 multicasting message to group $G = \{n_1, n_2\}$

(b) Node m multicasting message to group $G = \{n_1, n_2\}$

Figure 6: Multicast Operation

In our example, the requests are bounded to two levels. All higher-level logical servers need not be updated.

6 Quantitative Analysis: AMDLM vs. MDLM

In this section we study the performance of *AMDLM* compared to MDLM, the straightforward multicast extension of DLM presented in Section 4.1. The nature of *AMDLM* enables us to conduct an analysis not possible with other multicast algorithms such as MAODV [RP99] and ODMRP [LCG99].

6.1 Location Servers

6.1.1 Total Number of Location Servers

The number of required location servers has a direct impact on overhead. Given at least one group member, MDLM assigns 4^m uniformly distributed location servers, where m is the desired density.

The total number of location servers for *AMDLM* dynamically grows and shrinks with respect to the number and distribution of multicast group members. The upper bound at any given time for the total number of location servers is calculated as follows. It is equal to the sum across levels, of the number of $level_k$ regions containing at least one group member and *not* belonging to the same $level_{k+1}$ region, multiplied by four: $4\Sigma_{k=1}^{k_{max}} \mid \{R_k \mid \exists n \in R_k, n \in G\}$, where k_{max} is the highest level. This result is an upper bound since we do not subtract from the result overlapping location servers across different levels, implied by Property 2 of Section 5.3.

6.1.2 Join and Leave Operations

Let n_i be the node desiring to join or leave a group G and n_j any member of G. When n_i joins or leaves G, MDLM requires all 4^m location servers to be updated. *AMDLM* requires $4k$ location servers to be updated, where k is the smallest region containing n_i and another member of the group: k = smallest l such that $R_l^{n_i} = R_l^{n_j}$. Upon the first *join* (and last *leave*) all $4k_{max}$ servers are updated.

6.1.3 Mobility

Let n_i be the mobile node and k be the largest $level_k$ region traversed by n_i upon moving. Both MDLM and *AMDLM* limit the number of location servers to be notified of n_i's new position. *AMDLM* requires at most $2*4k$ location servers to be updated. Indeed, *AMDLM* handles mobility as combined *join* and *leave* operations (n_i rejoins the group upon arriving to destination and sends a query to its previous $level_0$ region to issue a *leave* query on its behalf).

MDLM requires $4k$ location servers to be updated with n_i's new position. Note that for MDLM k is determined by the density m of location servers, whereas for *AMDLM* k is determined by the total number of $level_0$ regions in the grid. MDLM does not specify how information is removed from the location servers responsible for maintaining n_i's position before having moved.

6.2 Total Distance Cost

We assume that there is sufficient connectivity in the network in order for any two adjacent $level_0$ regions to communicate. An upper bound on the total distance traveled by the *join* and *leave* operations may now be derived for *AMDLM* based on the number of levels[8]. For sake of simplicity, we consider the Manhattan distance.

Since a $level_k$ region contains 4^k $level_0$ regions, the upper bound δ on the distance traveled within a $level_k$ region is $\delta(k) = 2\sqrt{4^k} - 2 = 2^{k+1} - 2$ $level_0$ regions (from one corner to the diagonally opposite one). To obtain the upper bound on the distance, we multiply δ by the number of queries required for each relevant operation, per level, for both algorithms (similarly to Section 6.1.1).

join/leave : $4\Sigma_{l=1}^{k}\delta(l)$, mobility : $2(4\Sigma_{l=1}^{k}\delta(l)) + \delta(l)$

6.3 Discussion

The recurring dilemma for a location management service in MANETs is the dissemination and maintenance cost of location information versus the accessibility and quality of the information upon retrieval. *AMDLM* addresses this dilemma in the context of multicast by concentrating the effort of maintaining location servers nearby multicast group members, while offering minimal location service in regions with little or no group members.

MDLM's uniform location server distribution, while a reasonable assumption for unicast routing, does not offer the flexibility needed for an efficient access to multicast group information without the overwhelming overhead of maintaining the same quality of service across the entire area of the grid.

[8] Without the assumption about density, we would have been bound by theoretic results obtained for the underlying geographic forwarding layer [KWZ02].

7 Qualitative Comparison Between the Different Approaches for MANET Multicasting

In this section we conduct a qualitative analysis of the performance of *AMDLM*, MAODV and ODMRP under different factors based on the core design of each algorithm.

The main design choices for the underlying group maintenance structures in MANET multicast algorithms are *tree* and *mesh* (See Table 1). We compare *AMDLM* against the main representant of each approach, MAODV [RP99] and ODMRP [LCG99]. All these protocols are on-demand. For the reader unfamiliar with MAODV and ODMRP, a presentation may be found in Appendix A.

7.1 No Mobility

We first examine the behavior of the various protocols for static networks — nodes are not mobile.

7.1.1 Managing concentrated group members

ODMRP does not maintain multicast group membership when no node is interested in multicasting, generating no network activity whatsoever. Both MAODV and AMDLM are expected to exhibit similar behavior in terms of multicast group management. Nevertheless, AMDLM generates additional overhead, since it forwards group member location information to a bounded number of designated location servers.

7.1.2 Managing scattered group members

In absence of multicast senders, network activity for ODMRP is non-existent. MAODV actively maintains a multihop tree connecting multicast group member nodes. For few nodes scattered over a large geographic area, the number of non-member nodes in required to maintain the multicast tree greatly outnumbers the number of member nodes. In case of numerous group members, MAODV generates important overhead, since it constantly maintains a multicast tree connecting all members and periodically generates *group hello* packets. AMDLM assigns a bounded number of location servers responsible for managing multicast group membership (See Section 6). A judicious grid decomposition adapted to the number of nodes is required to achieve optimal overhead.

7.1.3 Multicasting

Since ODMRP does not actively manage group membership, mesh construction is required whenever nodes wish to multicast. Sender nodes create a mesh reaching multicast group members through periodic flooding. ODMRP does not scale well with respect to numerous senders or receivers (multicast group members) [KC02]. The reason for its robustness (mesh structure offering alternative paths) is that of its poor overhead performance. Indeed, ODMRP generates nearly as much overhead as pure flooding by disseminating data across the forwarding group (*scoped flooding*). With group members scattered over large regions, ODMRP will ultimately lead to network performance degradation similar to the broadcast storm problem discussed in [NTCS99].

In MAODV and AMDLM, multicast group members wishing to multicast have immediate access to routing information. For non-member nodes, MAODV requires flooding in order to reach a node on the multicast tree to obtain routing information. AMDLM induces the least overhead among the protocols, since flooding is not required at any stage. Routing information may at any time be retrieved from the logical servers referenced by the hash function.

7.2 Mobile Nodes

We now introduce the factor of mobility to the analysis. The main factor on performance is the frequent link breakages caused by mobility (node crashes may therefore be considered as a particular form of mobility).

7.2.1 MAODV

Group members are reachable through a bidirectional multicast tree, providing a single path composed of critical links to reach multicast group members. Node mobility increases the occurrence of link breakages, damaging MAODV's tree structure and leading to reduced multicast efficiency. Furthermore, the tree-based approach may even generate great overhead by incessantly reacting to broken links and initiating repairs. The more scattered the group members, the greater the chance for link breakages to occur. Even a single link breakage may obstruct the path to group members and force MAODV to repair the tree. The physical tree structure therefore proves highly fragile.

7.2.2 ODMRP

It has been shown in [LSH+00, KC02] that an increase in mobility has little impact on the efficiency of the algorithm. These results may be explained by two key characteristics of ODMRP. Firstly, ODMRP periodically recreates routes to destination nodes while sender nodes have packets to multicast, maintaining route freshness (although at the expense of increased overhead). Secondly, the underlying mesh structure is more robust than the tree approach by offering alternative paths to reach the multicast group members.

7.2.3 AMDLM

A location-based scheme offers relative independence from node mobility. Indeed, the logical tree is formed by nodes in geographic regions (logical servers) designated by the hash function. These geographic regions are fixed. Nevertheless, the time required for the relevant logical servers to be updated through geographic forwarding increases the latency of the multicast algorithm, i.e. the overall time required to locate and reach multicast group members. Furthermore, packet loss may occur if the location information in the logical servers is stale.

7.3 Additional Factors

AMDLM uses the logical servers not only for storing group membership but also for forwarding packets to group members. The decision to use logical servers for routing is to avoid round-trip delays to obtain the position of the group members, reducing overall latency. The greater number of packets transiting through the geographic regions responsible for managing group membership may however cause possible bottlenecks under high network traffic.

MAODV and ODMRP are independent from geographic concerns and may seamlessly function under reasonable arbitrary node distribution. AMDLM on the other hand expects to find one or more nodes in the regions designated by the hash function to manage multicast group membership. AMDLM therefore requires a node distribution tailored to the hash function (e.g. the hash function presented in Appendix B assumes a uniform node distribution).

7.4 Summary

By adopting a significantly different approach for multicast group management, *AMDLM* offers greater robustness and reliability than the tree-based approach of MAODV without the associated overhead induced by the mesh-based approach of ODMRP.

8 Summary and Future Work

To the best of our knowledge, we have presented the first location service specifically tailored for multicasting in MANETs. We have additionally specified *AMDLM*, a multicast algorithm built on top of the location service. The advantages of such an approach over multicast algorithms relying on a tree, DAG or mesh approach is the absence of a multihop data-structure connecting the source and members of the multicast groups. Such data-structures are indeed less robust to frequent link breakages that occur in the highly dynamic environment of mobile ad hoc networks.

Designing an efficient location service in the context of multicast within MANETs is a difficult task. To the contrary of unicast routing, the location service must efficiently maintain information regarding a *set* of scattered nodes associated with each group. Maintaining accessible multicast group membership across the network may quickly lead to high communication overhead, reducing performance and wasting limited resources. *AMDLM* addresses the dilemma of the location service overhead versus multicast group information accessibility by *dynamically* adjusting the number and density of the location servers.

For future work, we intend to compare *AMDLM* through simulation to MAODV [RP99] and ODMRP [LCG99] in order to fully comprehend the performance of each approach in various situations and validate the qualitative analysis presented in Section 7.

References

[BCST00] S. Basagni, I. Chlamtac, V. R. Syrotiuk, and R. Talebi. On-demand location aware multicast (OLAM) for ad hoc networks. In *Proceedings of the IEEE WCNC 2000*, Chicago, IL, Sep 2000.

[BCSW98] Stefano Basagni, Imrich Chlamtac, Violet R. Syrotiuk, and Barry A. Woodward. A distance routing effect algorithm for mobility (DREAM). In *Proceedings of the 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM-98)*, pages 76–84, New York, October 1998. ACM Press.

[CBW+02] Tracy Camp, Jeff Boleng, Brad Williams, Lucas Wilcox, and William Navidi. Performance Comparison of Two Location Based Routing Protocols for Ad Hoc Networks. In *IEEE INFOCOM 2002*, New York, NY, June 23–27 2002.

[CGZ98] C.-C. Chiang, M. Gerla, and L. Zhang. Forwarding Group Multicast Protocol (FGMP) for Multihop, Mobile Wireless Networks. *Baltzer Cluster Computing*, 1(2):187–196, Dec 1998.

[CLP+02] Christine T. Cheng, Howard L. Lemberg, Sumesh J. Philip, Eric van den Berg, and Tao Zhang. SLALoM: A scalable location management scheme for large mobile ad-hoc networks. In *Proceedings of IEEE Wireless Communications and Networking Conference (WCNC 2002)*, Orlando, Florida, Mar 2002.

[DS01] Vijay Devarapalli and Deepinder Sidhu. MZR: A multicast protocol for mobile ad hoc networks. In *IEEE International Conference on Communications (ICC)*, Helsinki, Finland, June 2001.

[HB03] M. Heissenbüttel and T. Braun. Ants-based routing in large scale mobile ad-hoc networks. In *The Proceedings of the Kommunikation in verteilten Systemen (KiVS03)*, Leipzig, Germany, March 2003.

[HOTV99] Christopher Ho, Katia Obraczka, Gene Tsudik, and Kumar Viswanath. Flooding for reliable multicast in multi-hop ad hoc networks. In *Proceedings of the 3rd International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, pages 64–71, Seattle, WA, August 1999.

[HXG02] X. Hong, K. Xu, and M. Gerla. Scalable routing protocols for mobile ad hoc networks. *IEEE Network*, July/August 2002.

[JC98] Lusheng Ji and M. Scott Corson. A Lightweight Adaptive Multicast Algorithm. In *Proceedings of the IEEE GLOBECOM'98*, volume 2, pages 1036–1042, Sydney, Australia, November 1998.

[JJ01] Jorjeta G. Jetcheva and David B. Johnson. Adaptive Demand-Driven Multicast Routing in Multi-Hop Wireless Ad Hoc Networks. In *Proceedings of the ACM Symposium on Mobile Adhoc Networking and Computing (MOBIHOC 2001)*, pages 33–44, Long Beach, CA, Oct 2001.

[JT87] John Jubin and Janet D. Tornow. The DARPA packet radio network protocol. 75(1):21–32, January 1987.

[KC02] T. Kunz and E. Cheng. On-demand multicasting in ad-hoc networks:comparing AODV and ODMRP. In *22nd International Conference on Distributed Computing Systems (ICDCS '02)*, pages 453–454, Washington - Brussels - Tokyo, July 2002. IEEE.

[KK00] Brad Karp and H. T. Kung. Greedy Perimeter Stateless Routing for Wireless Networks. In *Sixth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom 2000)*, pages 243–254, Boston, MA, August 2000.

[KWZ02] Fabian Kuhn, Roger Wattenhofer, and Aaron Zollinger. Asymptotically optimal geometric mobile ad-hoc routing. In *Proceedings of the 6th international workshop on Discrete algorithms and methods for mobile computing and communications*, pages 24–33, Atlanta, Georgia, USA, 2002. ACM Press.

[LCG99] S.-J. Lee, C. Chiang, and M. Gerla. On-demand multicast routing protocol. In *Proceedings of the IEEE WCNC'99*, pages 1298–1304, New Orleans, USA, Sep 1999.

[LJC+00] Jinyang Li, John Jannotti, Douglas S. J. De Couto, David R. Karger, and Robert Morris. A scalable location service for geographic ad hoc routing. In *Proceedings of the 6th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, August 2000.

[LSH+00] Sung-Ju Lee, William Su, Julian Hsu, Mario Gerla, and Rajive Bagrodia. A performance comparison study of ad hoc wireless multicast protocols. In *Proceedings of IEEE Infocom 2000*, pages 565–574, Tel-Aviv, Israel, Mar 2000. IEEE.

[LTM99] Mingyan Liu, Rajesh R. Talpade, and Anthony McAuley. AMRoute: Adhoc Multicast Routing Protocol. Technical Report 99, The Institute for Systems Research, University of Maryland, 1999.

[MFWL03] Martin Mauve, Holger Fï¿½er, Jrg Widmer, and Thomas Lang. Position-based multicast routing for mobile ad-hoc networks. Technical Report TR-03-004, University of Mannheim, 2003.

[MGLA01] E. L. Madruga and J. J. Garcia-Luna-Aceves. Scalable Multicasting: The Core Assisted Mesh Protocol. *ACM/Baltzer Mobile Networks and Applications, Special Issue on Management of Mobility in Distributed Systems*, 6(1), 2001.

[MWH01] M. Mauve, J. Widmer, and H. Hartenstein. A survey on position-based routing in mobile ad-hoc networks. *IEEE Network Magazine*, 15(6):30–39, Nov 2001.

[NTCS99] Sze-Yao Ni, Yu-Chee Tseng, Yuh-Shyan Chen, and Jang-Ping Sheu. The broadcast storm problem in a mobile ad hoc network. In *Proceedings of the Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking*, pages 151–162, August 1999.

[OTV01] Katia Obraczka, Gene Tsudik, and Kumar Viswanath. Pushing the limits of multicast in ad hoc networks. In *Proceedings of the IEEE 21st International Conference on Distributed Computing Systems (ICDCS'01)*, Mesa, Arizona, Apr 2001.

[PRD01] Charles E. Perkins, Elizabeth M. Royer, and Samir R. Das. Ad hoc On-Demand Distance Vector (AODV) Routing. Internet Draft (draft-ietf-manet-aodv-09.txt), November 2001. Work in Progress.

[RP99] Elizabeth M. Royer and Charles E. Perkins. Multicast operation of the ad-hoc on-demand distance vector routing protocol. In *Proceedings of the fifth annual ACM/IEEE international conference on Mobile computing and networking*, pages 207–218. ACM Press, 1999.

[SSB99] P. Sinha, R. Sivakumar, and V. Bharghavan. MCEDAR: Multicast core extraction distributed ad-hoc routing. In *Proceedings of the Wireless Communications and Networking Conference (WCNC 1999)*, pages 1313–1317, New Orleans, LA, 1999.

[Sto02] I. Stojmenovic. Position based routing in ad hoc networks. *IEEE Commmunications Magazine*, 40(7):128–134, Jul 2002.

[TGB00] C.-K. Toh, Guillermo Guichal, and Santithorn Bunchua. ABAM: On-Demand Associativity-Based Multicast Routing for Ad hoc Mobile Networks. In *Proceedings of the IEEE Vehicular Technology Conference, VTC 2000*, pages 987–993, Boston, MA, Sep 2000.

[WS01] Seung-Chul M. Woo and Suresh Singh. Scalable routing protocol for ad hoc networks. *Wireless Networks*, 7(5):513–529, 2001.

[WT99] C. W. Wu and Y. C. Tay. AMRIS: A Multicast Protocol for Ad hoc Wireless Networks. In *Proceedings of the IEEE Military Communications Conference (MILCOM)*, pages 25–29, Atlantic City, NJ, November 1999.

[XLN01] Yuan Xue, Baochun Li, and Klara Nahrstedt. A scalable location management scheme in mobile ad-hoc networks. In *In in Proceedings of the 26th IEEE Annual Conference on Local Computer Networks (LCN 2001)*, pages 102–111, Tampa, Florida, Nov 2001.

[YKV99] Y.B-Ko and N.H. Vaidya. Geocasting in mobile ad hoc networks: Location-based multicast algorithms. In *Proceedings of the IEEE WMCSA'99*, pages 101–110, New Orleans, LA, Feb 1999.

Appendices

A Brief Presentation of the MAODV and ODMRP Multicast Algorithms

This appendix presents the MAODV [RP99] and ODMRP [LCG99] MANET multicast algorithms.

A.1 MAODV

MAODV is an *on-demand tree*-based multicast protocol based on the AODV [PRD01] unicast routing algorithm. MAODV uses a bidirectional tree for multicast group membership management (one tree per group). The tree is composed of group members and connecting nodes. Each tree contains a *leader* — usually the first node to join the group — responsible for maintaining group freshness through group *hello* messages.

Multicast group joins and leaves are handled explicitly in MAODV. When a node desires to join a group, it floods a *route request* packet across the network using an expanding ring search technique[9]. Intermediate nodes on the path cache a reverse route to the joining node according to previous hop information. Nodes on the multicast tree having received the flooded packet unicast back a *route reply* packet along the route to the joining node. After a *discovery* waiting period intended to collect *route reply* packets, the joining node attaches itself to the tree by unicasting a *multicast activation* packet to the closest (in terms of number of hops) node of the multicast tree.

While any member may leave its group(s), only leaf nodes are physically removed from the multicast tree. Other nodes are still necessary to maintain the tree structure. A multicast group member node leaves the tree by unicasting a *prune* request to its 1-hop neighbor on the tree. Upon receiving the request, the node on the tree removes the leaving node from its next hop table.

Multicasting is possible for both member and non-member nodes. For non-member nodes, a route-discovery phase similar to the multicast group *join* operation is conducted. Upon receiving a *route request* packet, any node with routing knowledge to the group responds to the sender with the routing information.

MAODV also addresses route maintenance through a tree repair procedure required to handle link breakages. Links on the multicast tree are considered broken if no packet (regular or *hello*) has been received from a neighbor for a timeout interval. If a broken link is detected, the disconnected node on the leaderless subtree attempts to reconnect by initiating a *local* repair through a *route request* with limited *TTL*. If reconnection is not possible (e.g. network partition), the disconnected node becomes the new group leader and a new independent tree is formed.

A.2 ODMRP

ODMRP [LCG99] is an *on-demand mesh*-based multicast algorithm independent of an underlying unicast routing protocol. The mesh is composed of overlapping *per sender* trees. Nodes in the mesh constitute a *forwarding group* over which multicast packets are flooded (i.e. *scoped* flooding). The mesh structure helps ODMRP achieve increased robustness to mobility and node failure through the multiple alternative paths, yet at the cost of increased overhead.

In ODMRP, multicast group members are called *receiver* nodes. Receivers locally track their own group membership and do not directly broadcast packets. *Senders* on the other hand are nodes desiring to multicast messages and are required to actively join the multicast group. To do

[9] If the node happens to know a route leading to the group leader, it may directly unicast the request to the leader.

so, a sender floods a `join` request packet throughout the entire network. Intermediate nodes on the path cache a reverse route to the sender according to previous hop information. When the packet reaches receiver nodes, each one of the *receivers* selects a single 1-hop neighbor [10] from which it desires to receive the sender's multicast data stream and sends it a `join reply` packet. Nodes receiving the `join reply` packet repeat the procedure until the initial sender is reached, ultimately creating a multicast tree rooted at the *sender*. The mesh is composed of union of all per-sender trees. Nodes continuously rejoin the group as long as they have data to multicast. In this sense, ODMRP is more proactive than MAODV, which only triggers activity upon a topology change on the multicast tree.

No control message is sent when a node desires to leave a group (*soft-state* approach). When senders stop emitting periodical `join` requests for the group, the multicast tree rooted at the sender will naturally dissolve after a fixed timeout[11].

A node desiring to multicast a message to a group must beforehand create a mesh containing the receivers, by means of the procedure described above. The source node then floods the multicast message across the mesh.

No dedicated route maintenance or repair procedure is required by ODMRP, since the periodical broadcast of `join` requests by nodes desiring to multicast contributes to the continual reconstruction of the mesh.

B An MDLM Hash Function Specification

B.1 Addressing Scheme and Notations

Figure 7 presents the addressing scheme by which $level_0$ squares are identified. Two bits are required to identify one of the four squares composing a region of any level. Beginning at the highest-level region, an address is assigned counter-clockwise to each one of its four regions, starting from the bottom-left corner of the grid. The same scheme is repeated in a recursive manner by appending at each level two extra bits to the address until all of the k level regions have been assigned an address. Figure 7 shows the $level_0$ region fully identified by 101001.

We recall the definitions of $S_k^{n_i}(G)$ and $\overline{S}_k^{n_i}$. In Section 5.3 we have defined $S_k^{n_i}(G)$ the logical server(s) in a region R_k for node n_i member of group G. $\overline{S}_k^{n_i}$ is by definition the $level_k$ location server located in n_i's $level_{k-1}$ region.

B.2 Specification

We present in this section a specification for the hash function η verifying Properties 1 and 2 mentioned in Section 5.3. The addressing scheme and notations described in Appendix B.1 are assumed.

The address of the logical servers returned by η are the result of the concatenation of a *prefix*, *position* and *suffix*. Functions 1 and 2 make use of a `substring(text, s, c)` function, which returns the substring of `c` letters of string `text`, beginning at position `s`. We define $\mathbb{A}_n$ the set of binary strings of length n and $\mathbb{B}_n = \{\{b_1, b_2, b_3, b_4\} \mid b_1, b_2, b_3, b_4 \in \mathbb{A}_n\}$. We keep in mind that for a grid with k_{max} levels, the address size of the $2^{2(k_{max}-1)}$ $level_0$ regions is of $2(k_{max}-1)$ bits ($k_{max}-1$ bits for each of the x and y coordinates). Let $a \in \mathbb{A}_{2(k_{max}-1)}$, $l \in [0, k_{max}-1]$ a $level_0$ region address and $G \in \mathbb{N}$ a group ID.

[10] The 1-hop neighbor is selected from the neighbors from which it has received a `join` packet.

[11] ODMRP makes use of misleading terminology concerning the *join* operation, required by nodes desiring to multicast (senders) rather and not by multicast group members (receivers), which are generally passive.

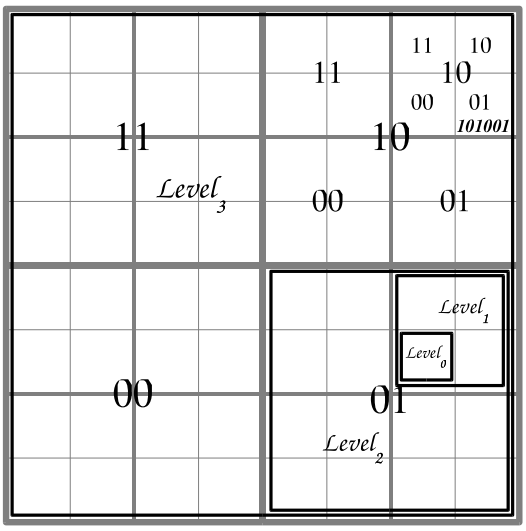


Figure 7: Grid Partitioning and Addressing Scheme

Function 1 (prefix) *Address of the enclosing* $level_{l+1}$ *region, obtained by taking the first* $2(k_{max} - l - 1)$ *bits of the requesting node's address.*

$prefix : \mathbb{A}_{2(k_{max}-1)} \rightarrow \mathbb{A}_{2(k_{max}-l-1)}$

$prefix(k_{max}, l, a) = substring(a, 1, 2(k_{max} - l - 1))$

Function 2 (position) 2 *bits indicating the four* $level_l$ *regions of* a.

$position : \mathbb{A}_{2(k_{max}-1)} \rightarrow \mathbb{B}_2$

$position(k_{max}, l, a) = (substring(a, 2(k_{max} - l - 1) + 1, 2) + i) mod 4,\ i = \{00, 01, 10, 11\}$

When $i = 00$, $\overline{S_l^{n_i}}(G)$ is returned.

In order to distribute fairly the responsibility of logical servers for each group G across the 2^{2l} $level_0$ regions, we define $suffix$ as being:

Function 3 (suffix) $suffix(G, l)$ $2(l-1)$ *bits designating a* $level_0$ *region to serve as a logical server within the* $level_l$ *region.*

$suffix :\ \mathbb{A}_{2(k_{max}-1)} \rightarrow \mathbb{A}_{2(l-1)}$

$$suffix(G, l) = \begin{cases} G \bmod 2^{2l} & \text{if } l > 1 \\ \text{empty string} & \text{otherwise} \end{cases}$$

We are now able to define the hash function η, through which the $level_0$ regions for the logical servers may be obtained.

Function 4 (η) *Returns the addresses of all logical servers for* $l \geq 1$

$\eta :\ \mathbb{A}_{2(k_{max}-1)} \rightarrow \mathbb{B}_{2(k_{max}-1)}$

$\eta(k_{max}, l, G, a) = prefix(k_{max}, l, a) \bullet position(k_{max}, l, a) \bullet suffix(l, G)$ *(where* $\bullet$ *is the concatenation of s*

By convention, $\eta(k_{max}, l, G, a)$ *invoked with* $l = 0$ *returns* a.

In other words, η will be of the form $\underbrace{Prefix}_{2(k_{max}-l-1)bits} \overbrace{Position}^{2bits} \underbrace{Suffix}_{2(l-1)bits}$.

Example 1 *A grid with $k = 4$ levels will have $2^{2(4-1)} = 64$ $level_0$ regions, each identified by a $2(4-1) = 6$ bit address. Let n_i be a node of address* 100101 *(cf. Figure 7) and G the multicast group of $ID = 5$. Table 2 illustrates how η is used to obtain n_i's logical servers for the different levels.*

Level	**1**	**2**	**3**
Prefix	1001	10	-
Position	01, 10, 11, 00	01, 10, 11, 00	10, 11, 00, 01
Suffix	-	01	0101
Address $\overline{S}$	100101	100101	100101
Address 1	100110	101001	110101
Address 2	100111	101101	000101
Address 3	100100	100001	010101

Table 2: Example for η for node n_i of address 100101 with $k = 4$ levels.

We now present proofs for how the specification of the hash function η verified the required properties cited in Section 5.3.

Property 1 (Partitioning Rule)
$k = 0$: For each node n_i member of G, there exists exactly one logical server of level 0 (denoted by $S_0^{n_i}$), which corresponds to region $R_0^{n_i}$.
$k > 0$: For each level $k > 0$ and for each node n_i, there exists exactly four logical servers of level k (denoted by $S_k^{n_i}$). Moreover, there is exactly one of these four servers $S_k^{n_i}$ in each of the four sub-regions region $R_{k-1}^{n_i}$ of region $R_k^{n_i}$.

Proof. The positions of the logical servers $S_k^{n_i}$ computed for a node n_i by the hash function are always located in region $R_k^{n_i}$ because, according to the definition of the *prefix* function (Function 1 of Section 5.4), n_i has the same $2(k_{max} - k - 1)$ first bits as all its $S_k^{n_i}$ servers. Moreover, the *position* function (Function 2 of Section 5.4) ensures that exactly one $S_k^{n_i}$ server is placed in each R_{k-1} regions since it enumerates all possible R_{k-1} regions by iterating over i ($i = 00, 01, 10, 11$). In the particular case when $k = 0$ the hash function η returns a (cf. Function 4), the address of the $level_0$ region containing n_i.

Property 2 (Sharing Rule)
$\forall k > 0$ and for two nodes n_i and n_j member of the same group, if $R_k^{n_i} = R_k^{n_j}$, then n_i and n_j share the same level k server(s).

Proof. Let assume $R_k^{n_i} = R_k^{n_j}$. According to the definition of the *prefix* function (Function 1 of Section 5.4), the addresses of n_i and n_j have the $2(k_{max} - l - 1)$ first bits in common. Thus we have $prefix(k_{max}, k, R_0^{n_i}) = prefix(k_{max}, k, R_0^{n_j})$. For the same level k, the 2 bits returned by the *position* function differ for n_i and n_j. But the 2-bits generated by n_i and n_j will refer anyway to the same positions but maybe in a different order. The *suffix* computed by n_i and n_j are obviously equal because the *suffix* definition (Function 3 of Section 5.4) only depends on the group G and the level k. This implies that the unions $\{\overline{S}_k^{n_i}(G)\} \cup S_k^{n_i}(G)$ and $\{\overline{S}_k^{n_j}(G)\} \cup S_k^{n_j}(G)$ contain the same regions if $R_k^{n_i} = R_k^{n_j}$

C AMDLM Algorithm

C.1 AMDLM Architecture

In this section we present the architecture and relationships between the various layers that are required by *AMDLM*. The *AMDLM* algorithm can be logically split into three different building blocks as depicted in Figure 8. At the lowest level, the algorithm relies on a *Geographic Forwarding* ([KWZ02, MWH01, Sto02]) layer, providing a geographic-based point-to-point unicast by means of a `geo-send` and corresponding `geo-receive` procedures. The *Geographic Forwarding* implementation is not described in this paper.

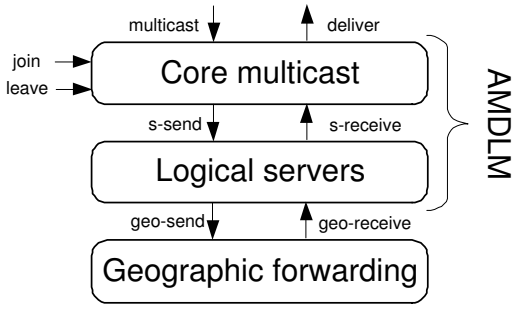


Figure 8: AMDLM Architecture

Above this unicast layer we provide the *logical servers* abstraction. This layer considers only communication links between *logical servers* and assumes that the physical terrain is divided into hierarchical regions as defined in Section 5.2. Its interface provides a communication primitive: `s-send(msg)`, that sends a message `msg` to a *logical server* S_k. Logical servers are fixed, distributed and hierarchical R_0 regions. They form a geographically static infrastructure that avoid costly routes maintenance operations and prevents nodes from tracking mobile recipients. All nodes in a R_0 region that acts as a *logical server* have the ability of offering the service. Practically, a node is arbitrarily singled out to take action (e.g. the node with the highest ID). This mechanism offers some kind of fault tolerance as long as at least one node is present in R_0.

The hierarchical distribution of the *logical servers* simplifies, in most cases, the group membership maintenance due to the mobility of group members. More precisely, whenever a mobile group member moves locally only the closest *logical servers* need to be updated. This is also true each time a new node joins a group G in the neighborhood of an other member. This feature reduces significantly the dissemination of membership informations throughout the whole network.

The third building block above the *logical servers* layer contains the *AMDLM* implementation. It relies on the communication environment provided by the *logical servers* abstraction. It allows a mobile node to multicast a message `msg` to a group G. Besides the `multicast(msg,`G`)` procedure a node can join a group G by calling the `join(`G`)` function. This operation will create or update membership informations in all necessary $S_k^{n_i}$ *logical servers*. Similarly, the *AMDLM* block exports a `leave(`G operation that allows node to leave a group G. As for mobility issues, if a `join(`G`)` or a `leave(`G`)` is invoked in the neighborhood of an other member of the same group G, then only a local update is required.

C.2 AMDLM Algorithmic Specification

The model, notations and addressing scheme used by the algorithms are described in Section 5.2, Appendix B and Appendix C.1. Algorithms 1, 2 and 3 describe the operations *join*, *leave* and *multicast* invoked on the *AMDLM* layer. Then, Algorithm 4 shows how the mobility is handled when a node moves from one R_0 region to another. Finally, Algorithm 5 gives the pseudo-code for the *Logical server* layer, namely the interaction with the *Geographic forwarding* layer upon the invocation of *s-send* procedure.

Algorithm 1 : AMDLM - Join

1: {Upon `join(G)` by a node $n_i \in S_0$:}
2:
3: `s-send(join,`k_{max}`,G,`$S_0^{n_i}$`)` to $S_0^{n_i}$;
4:
5: {Upon `s-receive(join,level,G,`$S_l^{n_j}$`)` by a node $n_i \in S_k$}
6:
7: **if** k == 0 **then**
8: $members_i = members_i \cup \{(G, n_j)\}$;
9: **end if**
10: $links_i = links_i \cup \{(G, S_l^{nj})\}$;
11: **if** $k \leq level \leq k_{max}$ **and** ($\mid members_i \mid = 1$ **or** $\mid members_i \mid = 1$) **then**
12: `s-send(join,level,G,`$S_k^{n_i}$`)` to $S_{k+1}^{n_i}$;
13: **end if**

Algorithm 2 : AMDLM - Leave

1: {Upon `leave(G)` by a node $n_i \in S_0$:}
2:
3: `s-send(leave,`k_{max}`,G,`$S_0^{n_i}$`)` to $S_0^{n_i}$; {n_i *acts as the* S_0 *server*}
4:
5: {Upon `s-receive(leave,level,G,`$S_l^{n_j}$`)` by a node $n_i \in S_k$}
6:
7: **if** k == 0 **then**
8: $members_i = members_i \setminus \{(G, n_j)\}$;
9: **if** $members_i = \emptyset$ **and** $k \leq level \leq k_{max}$ **then**
10: `s-send(leave,level,G,`$S_k^{n_i}$`)` to $S_{k+1}^{n_i}$;
11: **end if**
12: **else**
13: $links_i = links_i \setminus \{(G, S_l^{n_j})\}$;
14: **if** $links_i = \emptyset$ **and** $k \leq level \leq k_{max}$ **then**
15: `s-send(leave,level,G,`$S_k^{n_i}$`)` to $S_{k+1}^{n_i}$;
16: **end if**
17: **end if**

Algorithm 3 : AMDLM - Multicast

{Upon `multicast(m,G)` by a node $n_i \in S_0$:}

for all $(G, n_j) \in members_i$ **do**
  deliver m to n_j;
end for
for all $(G, S_k^{n_j}) \in links_i$ **do**
  `s-send(m,G,`$S_0^{n_i}$`)` to $S_k^{n_j}$;
end for
if $\exists\, k \leq k_{max} \mid k > max_l\{(G, S_l) \in links_i\}$ **then**
  `s-send(m,G,`$S_0^{n_i}$`)` to server $\overline{S}_{k_{max}-1}^{n_i}$;
end if

{Upon `s-receive(m,G,`$S_l^{n_j}$`)` by a node $n_i \in S_k$}

if k = 0 **then**
  for all $(G, n_h) \in members_i$ **do**
    deliver m to n_h;
  end for
else if $links_i \cap \{(G, S_{k-1}^{n_h})\} \neq \emptyset$ **then**
  `s-send(m,G,`$S_k^{n_i}$`) to server` $S_{k-1}^{n_h}$;
end if
if $l \leq k < k_{max}$ **then**
  `s-send(m,G,`$S_k^{n_i}$`)` to $\overline{S}_{k+1}^{n_i}$;
end if

Algorithm 4 : Mobility management

{Upon node n_i has moved from S_0^{old} to $S_0^{n_i}$:}

{n_i must update $members_i$ and $links_i$ sets according to its new position}
if $members_i \cap \{(G, n_i)\} \neq \emptyset$ **then**
  level = minimum k such that $R_k^{R_0} = R_k^{R_0'}$;
  `s-send(join,level,G,`$S_0^{n_i}$`)` to $S_0^{n_i}$;
  `s-send(leave,level,G,`$S_0^{n_i}$`)` to S_0^{old};
end if

Algorithm 5 : Logical server algorithm

{Upon `s-send(content,G,`$S_k^{n_i}$`)` to a server $S_l^{n_j}$ by a node $n_i \in S_k$:}

$S_l = \eta(k_{max}, l, G, R_0^{n_j})$;
$S_k = \eta(k_{max}, k, G, R_0^{n_i})$;
for all $R_0 \in S_l$ **do**
  if $dist(R_0^{n_i}, R_0) < dist(R_0', R_0), \forall\, R_0' \neq R_0^{n_i} \in S_k$ **then**
    `geo-send(content)` to R_0;
  end if
end for

{Upon `geo-receive(content)` by a node $n_i \in S_k$ from a server S_j :}

`s-receive(content,`S_j`)`;